BBM 418 - Computer Vision Laboratory

HACETTEPE UNIVERSITY
Department of Computer Engineering

Programming Assignment-2
Spring 2018-2019
Dr. Nazli Ikizler Cinbis
TA. Ozge Yalcinkaya

Due Date: 23:59 pm on Monday, May 6th, 2019

Image Classification with Convolutional Neural Networks

In this assignment, you will get familiar with a more advance image representation and classification method. You will learn how to do a transfer learning by using CNNs and how to train a network for a specific problem. At the end, you will compare all the results and make comments on them. You will use pre-trained VGG-16 model and images from a scene dataset.

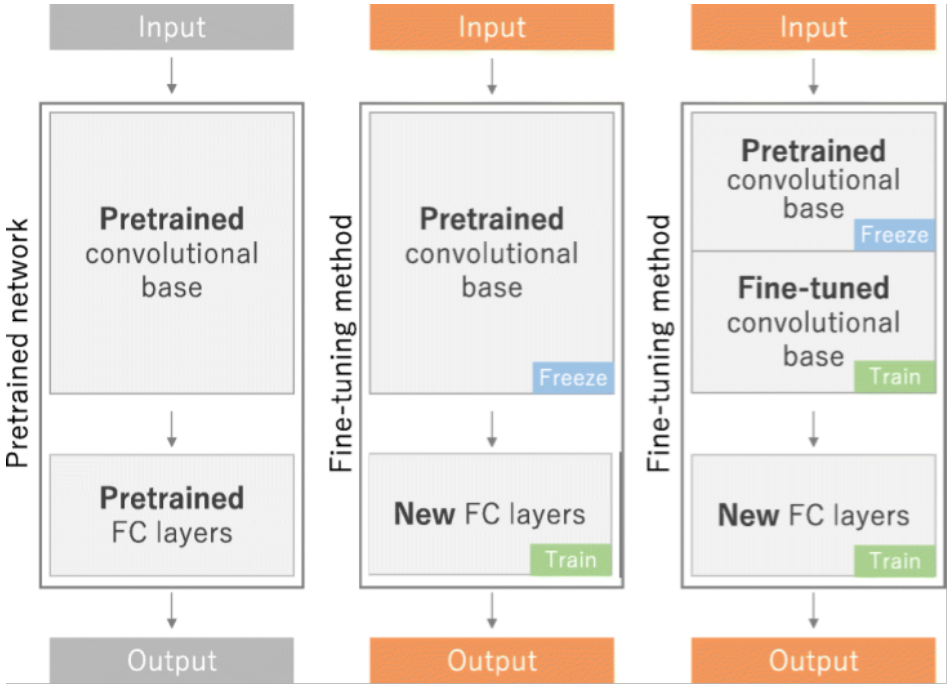

Figure 1: Overview of fine-tuning.

Dataset

The dataset has 10 scene categories and it is splitted into train, validation and test sets. You will use validation set only in Part 2 to monitor the learning in the network. Otherwise, you will give the scores over the test set. For each category you have 400 training, 25 validation and 25 test images.

PART 1 - Image Representation

Directly use the pre-trained VGG-16 model from torch library that is trained on ImageNet. Get the vectors of FC-7 after ReLU as features for each image in the dataset. In order to do this, you should run the network in evaluation mode so that **you are not training any weights**. You will just obtain the $1x4096$ vectors for each image from the specified layer by using a pre-trained network which is trained on ImageNet dataset.

Use one-vs-the-rest multi-class linear SVM for classification and get class based/total accuracies as defined previously in assignment 1. Also include a confusion matrix of the predictions, that indicates the confusions between classes. What do you observe about the types of errors the network makes?

Note: In short, you will use the network to only obtain the representations of the images in both train and test.

PART 2 - Fine-tuning

Now, you will fine-tune the given pre-trained VGG-16 network which you have loaded all the weights from the torch library. This network is trained on ImageNet dataset so you are not initializing the weights randomly, instead, you are using the pre-trained weights from this network. Freeze all the layers before training the network, except the FC layers. Consequently, the gradients will not be calculated for the layers except the ones that you are going to update (FC layers) over the pre-trained weights. However, since the number of classes is different for our dataset, you should modify the last layer of the network, FC-8, which the probabilities will be calculated on. Therefore, the weights will be randomly initialized for this layer.

What is fine-tuning? Why should we do this? Why do we freeze the rest and train only FC layers? Give your explanation in detail.

Bonus: Explore different experiments on training different number of layers such as training also the last conv layer and so on.

How to train a network?

- A network is trained by taking batches from the shuffled training set (explore PyTorch dataloader class which simplifies all training procedure for you). Therefore, you have to determine a **batch size**.
- While training your network, you should also use the validation set to monitor how your network generalizes the learning. To do this, you will set the network to evaluation mode and use the validation set as a test set during the training (e.g. after an epoch over the training set) and see the loss and accuracy (both for training and validation set).
- You read a batch of images from training data and train your network with it. It is called "an iteration". You have to repeat the same process until you use all your data which is in the number of "$\#training\_images/B$". This one pass on all of the training data is called "an epoch". You have to train your model for several epochs, which can be determined with **number of epochs**. In general, you can stop the learning epochs if the validation loss is not decreasing anymore. Research about "early stopping".
- The other important hyperparameter is the **learning rate**. You should determine a proper learning rate.

Give all of your results over the experiments and comment on them, give top-1 and top-5 accuracy of the test set with softmax probs. Use your network which had the best accuracy on the validation set to get the scores from the test set (run the network with the test set in evaluation mode). You should experiment with different batch sizes and learning rates. Monitor your loss graph among the epochs and try to observe the effects of the parameters (You can use Tensorboard plugin for PyTorch.) Use ADAM optimizer and Cross Entropy loss. You will get additional points for any other experiment/examination that you conduct.

Bonus: Try to add different regularizations to improve your results which will provide a generalization for your model to prevent overfitting. For example, adding dropout and weight decay.

Note: Please refer to the example script that we have discussed in the lab for further information (link is given in the Piazza post) But do not copy and paste this script.

PART 3

Repeat Part1 to get the features over your fine-tuned network (not the pre-trained ImageNet weights) and apply SVM again. Compare the results of Part1 and Part3. Explain why the results have changed.

The Implementation Details

1. You can write different scripts for each part.
2. You should pay attention to code readability such as comments, function/variable names and your code quality: 1) no hard-coding 2) no repeated code 3) cleanly separate and organize your code 4) use consistent style, indentation
3. Implement your code with Python 2.7-3 and use libraries from Anaconda. You can install any library that is not in Anaconda as well, such as OpenCV.
4. You should use the PyTorch as the deep learning framework. You can use Google Colab to run your experiments.

What should you write in the report?

- Give explanations for each step.
- Give experimental results, used parameters and comments on the results in detail.
- Give your model's loss plot and accuracy plot both for training and validation set during training.
- Put the results of different hyper-parameters (learning rate, batch size), the effect of them, with the loss/accuracy plots.
- A basic structure might be: 1) Introduction (what is the problem, how do you approach to this problem, what is the content of your report) 2) Implementation Details (the method you followed and details of your solution) 3) Experimental Results (all results for separate parts with different parameters and your comments on the results) 4) Conclusion (what are the results and what are the weaknesses of your implementation, in which parts you have failed and why, possible future solutions)
- You should write your report in LaTeX
- You should give visual results by using a table structure.

What to Hand In

Your submission format will be:

- README.txt *(give a text file containing the details about your implementation, how to run your code, the organization of your code, functions etc.)*
- code/ *(directory containing all your code)*
- report.pdf

Archieve this folder as **b<studentNumber>.zip** and submit to `https://submit.cs.hacettepe.edu.tr`.

Grading

The assignment will be graded out of 100:

- **25 (part 1):** CODE: 0 (no implementation), 5 (a partially correct solution), 10 (a correct solution) and REPORT: 15
- **50 (part 2):** CODE: 0 (no implementation), 15 (a partially correct solution), 25 (a correct solution) and REPORT: 25
- **25 (part 3):** CODE: 0 (no implementation), 5 (a partially correct solution), 10 (a correct solution) and REPORT: 15

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.